WEEK 6

Working with Real World Datasets

Rav Ahuja

© IBM Corporation. All rights reserved.

Working with CSV files

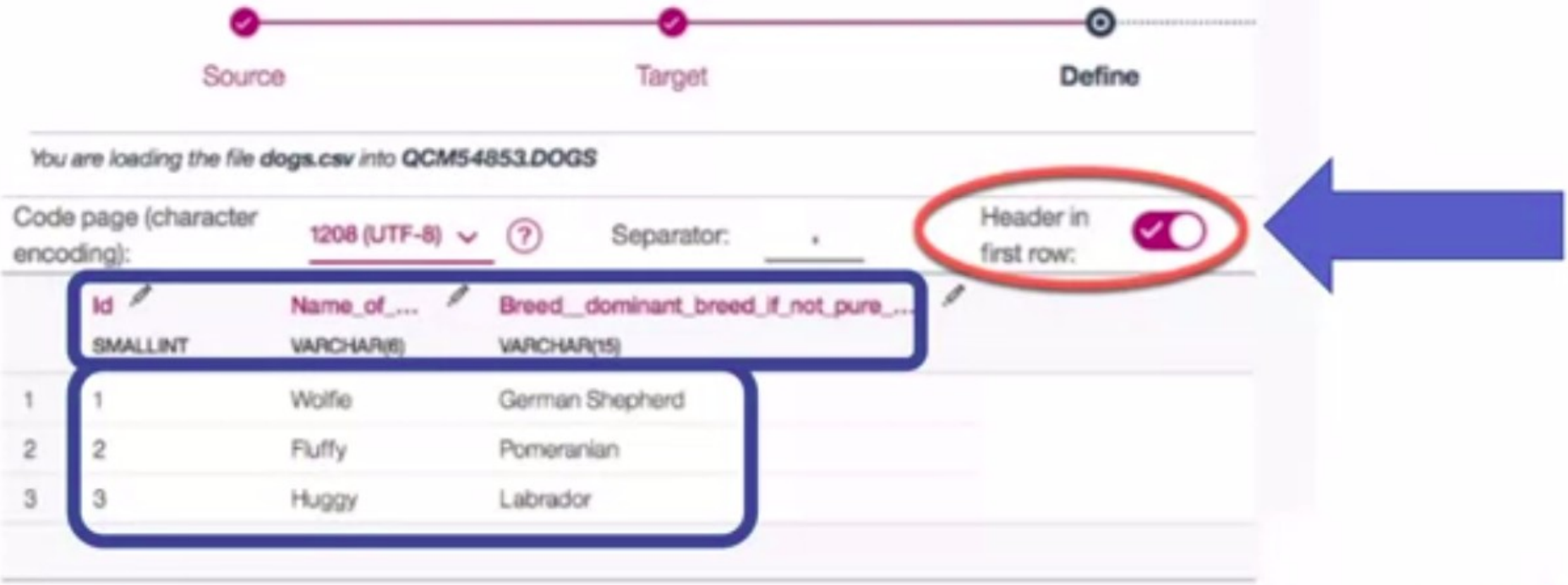
- Many real data sets are .CSV files
- .CSV: COMMA SEPARATED VALUES

- Example: DOGS.CSV

```
Id,Name of Dog,Breed (dominant breed if not pure breed)
1,Wolfie,German Shepherd
2,Fluffy,Pomeranian
3,Huggy,Labrador
```

Column names in first row

When header row in CSV file contains column names:



Querying column names with mixed (upper and lower) case

Retrieve Id column from DOGS table. Try:

```
select id from DOGS
```

If you run this query, you will get this error:

```
Error: "ID" is not valid in the context where it is
used.. SQLCODE=-206, SQLSTATE=42703, DRIVER=4.22.36
```

Use double quotes to specify mixed-case column names:

```
select "Id" from DOGS
```

Querying column names with spaces and special characters

By default, spaces are mapped to underscores:

	A
1	Name of Dog
2	

→ `Name_of_Dog`

Other characters may also get mapped to underscores:

```
select "Id", "Name_of_Dog",
"Breed__dominant_breed_if_not_pure_breed_"
from dogs
```

Breed (dominant breed if not pure breed)

Using quotes in Jupyter notebooks

First assign queries to variables:

```
selectQuery = 'select "Id" from dogs'
```

Use a backslash \ as the escape character in cases where the query contains single quotes:

```
selectQuery = 'select * from dogs
      where "Name_of_Dog"=\'Huggy\' '
```

Splitting queries to multiple lines in Jupyter

Use backslash "\" to split the query into multiple lines:

```
%sql select "Id", "Name_of_Dog", \
     from dogs \
     where "Name_of_Dog"='Huggy'
```

Or use %%sql in the first row of the cell in the notebook:

```
%%sql
select "Id", "Name_of_Dog",
     from dogs
     where "Name_of_Dog"='Huggy'
```

Restricting the # of rows retrieved

To get a sample or look at a small set of rows, limit the result set by using the LIMIT clause:

```
select * from census_data LIMIT 3
```

Getting Table and Column Details

Rav Ahuja

© IBM Corporation. All rights reserved.

Getting a list of tables in the database

Getting a list of tables in the database

DB2
SYSCAT.TABLES

SQL Server
INFORMATION_SCHEMA.TABLES

Oracle
ALL_TABLES or USER_TABLES

Getting a list of tables in the database

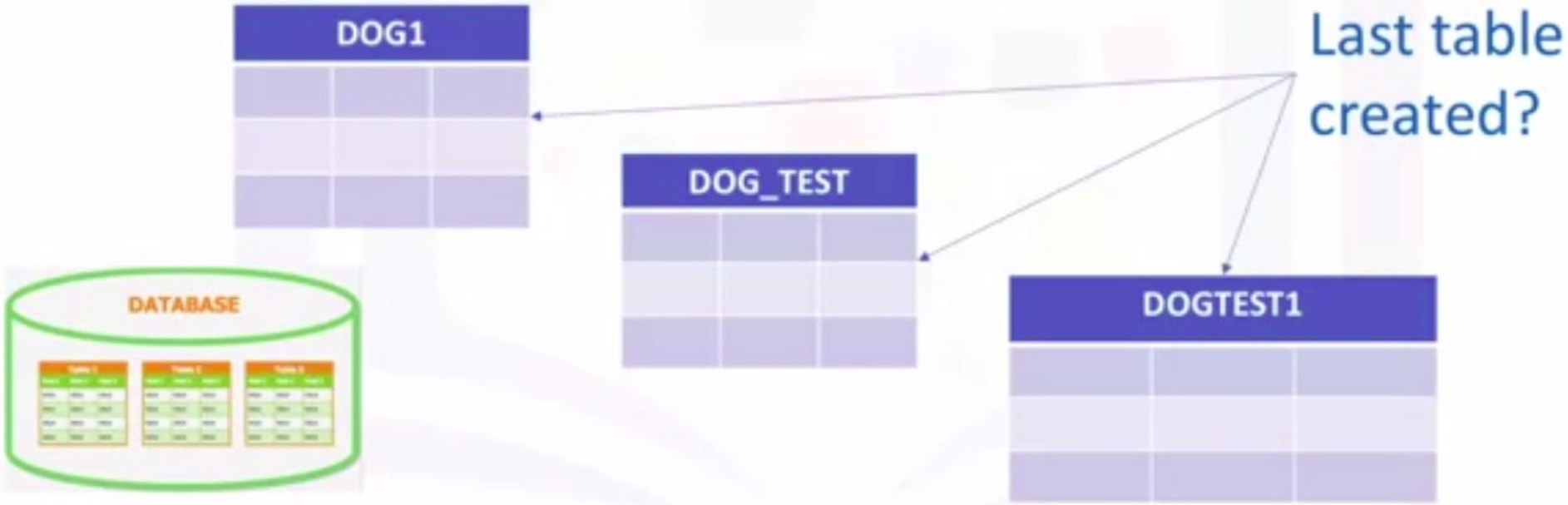
Query system catalog to get a list of tables & their properties:

```
select * from syscat.tables
```

```
select TABSCHEMA, TABNAME, CREATE_TIME
    from syscat.tables
    where tabschema= 'ABC12345'
```

Getting Table Properties

```
select * from syscat.tables
```



Getting Table Properties

```
select TABSCHEMA, TABNAME, CREATE_TIME
     from syscat.tables
     where tabschema='LCT12330'
```

```
[15]: %sql select TABSCHEMA, TABNAME, CREATE_TIME from SYSCAT.TABLES where TABSCHEMA='LCT12330'
 * ibm_db_sa://lct12330:***@dashdb-txn-sbox-yp-dal09-04.services.dal.bluemix.net:50000/BLUDB
Done.
```

[15]:

tabschema	tabname	create_time
LCT12330	PETRESCUE	2020-05-05 22:02:25.169368
LCT12330	BOOK	2020-02-20 22:37:50.351829
LCT12330	MEDALS	2020-02-22 02:50:54.841324
LCT12330	EMPLOYEES	2020-02-18 20:05:38.328981
LCT12330	JOB_HISTORY	2020-02-18 20:05:38.521203
LCT12330	JOBS	2020-02-18 20:05:38.700230
LCT12330	LOCATIONS	2020-02-18 20:05:39.050699
LCT12330	SCHOOLS	2020-04-16 22:06:58.174131
LCT12330	DEPARTMENTS	2020-04-15 20:01:16.429000
LCT12330	MCDONALDSNUTRITION	2020-04-23 16:40:35.205237

Getting a list of columns in the database

To obtain the column names query syscat.columns:

```
select * from syscat.columns
       where tabname = 'DOGS'
```

To obtain specific column properties:

```
select distinct(name), coltype, length
       from sysibm.syscolumns
       where tbname = 'DOGS'
```

Column info for a real table

In [12]:
```
%sql select distinct(name), coltype, length \
     from sysibm.syscolumns where tbname = 'CHICAGO_CRIME_DATA'
```

```
 * ibm_db_sa://qcm54853:***@dashdb-txn-sbox-yp-dal09-04.services.dal
Done.
```

Out[12]:

name	coltype	length
Arrest	VARCHAR	5
Beat	SMALLINT	2
Block	VARCHAR	35
Case_Number	VARCHAR	8
Community_Area	DECIMAL	4
Date	VARCHAR	22
Description	VARCHAR	46
District	DECIMAL	4
Domestic	VARCHAR	5
FBI_Code	VARCHAR	3

Back

Final Exam

Graded Quiz • 30 min • 10 total points

Due Sep 10, 11:59 PM IST

00:19:38 remaining

1. The SELECT statement is called a _____, and the output we get from executing the query is called a result set.

1 point

- ○ Table name
- ◉ Query
- ○ Function
- ○ Operator

2. Which of the following SQL statements will delete the customers where the Country is Italy?

1 point

- ○ DELETE FROM CUSTOMERS WHERE COUNTRY IS 'ITALY'
- ◉ DELETE FROM CUSTOMERS WHERE COUNTRY = 'ITALY'
- ○ DELETE COUNTRY 'ITALY' FROM CUSTOMERS
- ○ DELETE 'ITALY' FROM CUSTOMERS

3. What does the primary key of a relational table do?

1 point

3. What does the primary key of a relational table do? **1 point**

- ○ The primary key uniquely identifies each attribute in a table.
- ○ The primary key uniquely identifies each column in a table.
- ◉ The primary key uniquely identifies each row in a table.
- ○ The primary key uniquely identifies each relation in a table.

4. The basic categories of the SQL language based on functionality are _____ and Data Manipulation Language (DML). **1 point**

- ◉ Data Definition Language (DDL)
- ○ Data Entry Language (DEL)
- ○ Data Input Language (DIL)
- ○ Data Update Language (DUL)

5. When querying a table called Representative that contains a list of representatives and the state that they represent, which of the following queries will return the number of representatives from each state? **1 point**

5. When querying a table called Representative that contains a list of representatives and the state that they represent, which of the following queries will return the number of representatives from each state?

1 point

- ○ SELECT State, count(State) FROM Representative
- ◉ SELECT State, count(State) FROM Representative GROUP BY State
- ○ SELECT distinct(State) FROM Representative
- ○ SELECT State, distinct(State) FROM Representative GROUP BY State

6. You want to retrieve a list of cities in a state that have between 10,000 and 20,000 residents. Which clause would you add to the following SQL statement: **SELECT City, Residents FROMState**

1 point

- ○ WHERE Residents 10000 – 20000
- ○ WHERE Residents ARE BETWEEN 10000 AND 20000
- ◉ WHERE Residents BETWEEN 10000 AND 20000
- ○ WHERE Residents IN (10000, 20000)

7. Which of the following queries will retrieve the LOWEST value of PRICE in a table called PRODUCTS?

1 point

- ○ SELECT MAX(PRICE) FROM PRODUCTS
- ○ SELECT LOWEST(PRICE) FROM PRODUCTS
- ◉ SELECT MIN(PRICE) FROM PRODUCTS
- ○ SELECT LEAST(PRICE) FROM PRODUCTS

8. Which of the following queries will retrieve the PRODUCT NAME that has the highest price?

1 point

- ◉ SELECT PRODUCT_NAME FROM PRODUCTS WHERE UNIT_PRICE = (SELECT MAX(UNIT_PRICE) FROM PRODUCTS)
- ○ SELECT MAX(UNIT_PRICE) FROM PRODUCTS
- ○ SELECT PRODUCT_NAME FROM PRODUCTS WHERE UNIT_PRICE IS HIGHEST
- ○ SELECT PRODUCT_NAME FROM PRODUCTS WHERE UNIT_PRICE = MAX

9. A database cursor is a control structure that;

1 point

Back

Final Exam

Graded Quiz • 30 min • 10 total points

Due Sep 10, 11:59 PM IST

00:19:24 remaining

9. A database cursor is a control structure that;

1 point

- ○ Does not allow you to update records within a database
- ○ Does not allow communication with a database
- ◉ Enables traversal over the records in a database
- ○ Does not allow you to create tables

10. Cell magics: start with a double %% sign and apply to the entire cell. (T/F)

1 point

- ○ False
- ◉ True

Coursera Honor Code Learn more

☐ I, **Tushar Raha**, understand that submitting work that isn't my own may result in permanent failure of this course or deactivation of my Coursera account.

Assignment Preparation: Working with real-world data sets and built-in SQL functions

Assignment

Final Exam

Course Wrap-up

Reading: Congratulations & Next Steps
2 min

Congratulations on completing this Databases and SQL for Data Science with Python course! We hope you enjoyed it.

This course is part of:

- IBM Data Analyst Professional Certificate
- Applied Data Science Specialization
- IBM Data Science Professional Certificate
- IBM Data Engineering Foundations Specialization
- IBM Data Engineering Professional Certificate

Those of you following a Data Engineering track must also complete the Honors module in this course. It contains information on more advanced SQL techniques that will be useful to you as a Data Engineer.

As a next step, you can explore other courses in these programs, starting with:

- Data Engineering and Data Analytics tracks: Data Analysis with Python
- Data Engineering track: NoSQL Fundamentals: Introduction to NoSQL Databases

We encourage you to leave your feedback and rate the course.

Good luck!

About this Honors module

Reading: About this Honors module
2 min

Views, Stored Procedures, and Transactions

JOIN Statements

Quiz and Assignment for Advanced SQL

About this Honors module

This module covers some advanced SQL techniques that will be useful for Data Engineers. If you are following the Data Engineering track, you **must** complete this module.

This module is optional for those of you following the Data Science or Data Analyst tracks. If you choose to complete it, you will be awarded an Honors level certificate.

In this module, you will learn how to form more powerful queries by using advanced techniques like views, transactions, stored procedures, and joins.

Mark as completed

SKILLS NETWORK

Views

IBM Developer

Objectives

After watching this video, you will be able to:

- Define a view
- Describe when to use a view
- Explain the syntax for creating a view

What is a view?

EMP_ID	F_NAME	L_NAME	SSN	B_DATE	SEX	ADDRESS	JOB_ID	SALARY	MANAGER_ID	DEP_ID
E1001	John	Thomas	123456	1976-01-09	M	5631 Rice, ...	100	100000	30001	2
E1002	Alice	James	123457	1972-07-31	F	980 Berry Ln,	200	80000	30002	5
E1003	Steve	Wells	123458	1980-08-10	M	291 Springs,	300	50000	30002	5
E1004	Santosh	Kumar	123459	1985-07-20	M	511 Aurora,	400	60000	30004	5

What is a view?

EMP_ID	F_NAME	L_NAME	ADDRESS	JOB_ID	MANAGER_ID	DEP_ID
E1001	John	Thomas	5631 Rice, ...	100	30001	2
E1002	Alice	James	980 Berry Ln,	200	30002	5
E1003	Steve	Wells	291 Springs,	300	30002	5
E1004	Santosh	Kumar	511 Aurora,	400	30004	5

- Can include specified columns from multiple base tables and existing views
- Once created, can be queried like a table
- Only the definition of the view is stored, not the data

When should you use a view?

Views can:

- Show a selection of data for a given table
- Combine two or more tables in meaningful ways
- Simplify access to data
- Show only portions of data in the table

Example:

- Create a view to display non-sensitive data from the Employees table

CREATE VIEW statement

```
CREATE VIEW <view name> (<column_alias_1>,
<column_alias_2>, … <column_alias_n>)
  AS SELECT <column_1> , <column_2>, … <column_n>
  FROM <table name>
  WHERE <predicate>;
```

CREATE VIEW statement

```
CREATE VIEW EMPINFO (EMP_ID, FIRSTNAME, LASTNAME, ADDRESS,
JOB_ID, MANAGER_ID, DEP_ID)
  AS SELECT EMP_ID, F_NAME, L_NAME, ADDRESS, JOB_ID,
MANAGER_ID, DEP_ID
  FROM EMPLOYEES;
```

```
SELECT * FROM EMPINFO
```

EMP_ID	FIRST NAME	LASTNAME	ADDRESS	JOB_ID	MANAGER_ID	DEP_ID
E1001	John	Thomas	5631 Rice, ...	100	30001	2
E1002	Alice	James	980 Berry Ln,	200	30002	5
E1003	Steve	Wells	291 Springs,	300	30002	5
E1004	Santosh	Kumar	511 Aurora,	400	30004	5

Working with views

```
CREATE VIEW EMPINFO (EMP_ID, FIRSTNAME, LASTNAME, ADDRESS,
JOB_ID, MANAGER_ID, DEP_ID)
  AS SELECT EMP_ID, F_NAME, L_NAME, ADDRESS, JOB_ID,
MANAGER_ID, DEP_ID
FROM EMPLOYEES
WHERE MANAGER_ID = '30002'
```

```
SELECT * FROM EMPINFO
```

EMP_ID	FIRST NAME	LAST NAME	ADDRESS	JOB_ID	MANAGER_ID	DEP_ID
E1002	Alice	James	980 Berry Ln,	200	30002	5
E1003	Steve	Wells	291 Springs,	300	30002	5

Working with views

```
DROP VIEW EMPINFO
```

EMP_ID	FIRST NAME	LAST NAME	ADDRESS	JOB_ID	MANAGER_ID	DEP_ID
E1002	Alice	James	980 Berry Ln,	200	30002	5
E1003	Steve	Wells	291 Springs,	300	30002	5

Summary

In this video, you learned that:

- Views include specified columns from multiple base tables and existing views
- Views can be queried like a table
- Only the definition of the view is stored, not the data
- The CREATE VIEW statement creates a view based on one or more tables or views

SKILLS NETWORK

Stored Procedures

IBM Developer

What is a stored procedure?

Objectives

After watching this video, you will be able to:

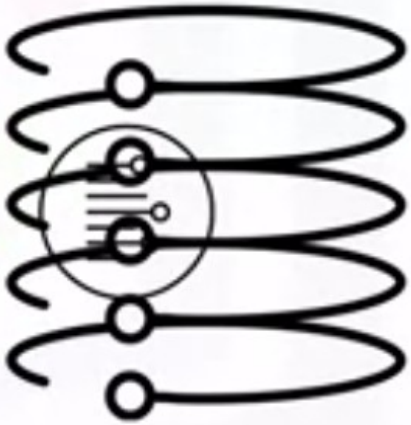
- Explain what a stored procedure is
- List the benefits of using stored procedures
- Describe how to create and use a stored procedure

What is a stored procedure?

- A set of SQL statements stored and executed on the database server

What is a stored procedure?

- A set of SQL statements stored and executed on the database server



What is a stored procedure?

- A set of SQL statements stored and executed on the database server
  - Write in many different languages
  - Accept information in the form of parameters
  - Return results to the client

CREATE PROCEDURE statement

```
CREATE PROCEDURE UPDATE_SAL (IN empNum CHAR(6), IN rating SMALLINT)
  LANGUAGE SQL
  BEGIN
    IF rating = 1 THEN
      UPDATE employees
        SET salary = salary * 1.10
        WHERE emp_id = empNum;
    ELSE
      UPDATE employee
        SET salary = salary * 1.05
        WHERE emp_id = empNum;
    END IF;
  END
```

Benefits of stored procedures

- Reduction in network traffic
- Improvement in performance
- Reuse of code
- Increase in security

CREATE PROCEDURE statement

```
CREATE PROCEDURE UPDATE_SAL (IN empNum CHAR(6), IN rating SMALLINT)
  LANGUAGE SQL
  BEGIN
    IF rating = 1 THEN
      UPDATE employees
        SET salary = salary * 1.10
        WHERE emp_id = empNum;
    ELSE
      UPDATE employee
        SET salary = salary * 1.05
        WHERE emp_id = empNum;
    END IF;
  END
```

Working with stored procedures

Call from:

- External applications
- Dynamic SQL statements

```
CALL UPDATE_SAL ('E1001', 1)
```

Summary

In this video, you learned that:

- Stored procedures are a set of SQL statements that execute on the server
- Stored procedures offer many benefits over sending SQL statements to the server
- You can use stored procedures in dynamic SQL statements and external applications

SKILLS NETWORK

ACID transactions

IBM Developer

Objectives

After watching this video, you will be able to:

- Explain what an ACID transaction is
- Give an example of an ACID transaction
- Describe commits and rollbacks

What is a transaction?

- Indivisible unit of work
- Consists of one or more SQL statements
- Either all happens or none

Transaction example

BankAccounts

AccountNumber	AccountName	Balance
B001	Rose	300
B002	James	13450
B003	Shoe Shop	124000
B004	Corner Shop	76000

ShoeShop

Product	Stock	Price
Boots	12	200
High heels	8	600
Brogues	10	150
Trainers	14	300

```
UPDATE BankAccounts
SET Balance = Balance-200
WHERE AccountName = 'Rose';
```

Transaction example

BankAccounts

AccountNumber	AccountName	Balance
B001	Rose	100
B002	James	13450
B003	Shoe Shop	124000
B004	Corner Shop	76000

ShoeShop

Product	Stock	Price
Boots	12	200
High heels	8	600
Brogues	10	150
Trainers	14	300

```
UPDATE BankAccounts
SET Balance = Balance+200
WHERE AccountName = 'Shoe Shop';
```

Transaction example

BankAccounts

AccountNumber	AccountName	Balance
B001	Rose	100
B002	James	13450
B003	Shoe Shop	124200
B004	Corner Shop	76000

ShoeShop

Product	Stock	Price
Boots	12	200
High heels	8	600
Brogues	10	150
Trainers	14	300

```
UPDATE ShoeShop
SET Stock = Stock-1
WHERE Product = 'Boots';
```

Transaction example

BankAccounts

AccountNumber	AccountName	Balance
B001	Rose	100
B002	James	13450
B003	Shoe Shop	124200
B004	Corner Shop	76000

ShoeShop

Product	Stock	Price
Boots	11	200
High heels	8	600
Brogues	10	150
Trainers	14	300

If any of these UPDATE statements fail, the whole transaction must fail

What is an ACID transaction?

Atomic

All changes must be performed successfully or not at all

Consistent

Data must be in a consistent state before and after the transaction

Isolated

No other process can change the data while the transaction is running

Durable

The changes made by the transaction must persist

ACID commands

- BEGIN
  - Start the ACID transaction
- COMMIT
  - All statements complete successfully
  - Save the new database state

```
BEGIN
      UPDATE BankAccounts
      SET Balance = Balance – 200
      WHERE AccountName = 'Rose’;

      UPDATE BankAccounts
      SET Balance = Balance + 200
      WHERE AccountName = 'Shoe Shop’;

      UPDATE ShoeShop
      SET Stock = Stock - 1
      WHERE Product = 'Boots’;
COMMIT
```

ACID commands

- BEGIN
  - Start the ACID transaction
- COMMIT
  - All statements complete successfully
  - Save the new database state
- ROLLBACK
  - One or more statements fail
  - Undo changes

```
BEGIN
      UPDATE BankAccounts
      SET Balance = Balance - 200
      WHERE AccountName = 'Rose';

      UPDATE BankAccounts
      SET Balance = Balance + 200
      WHERE AccountName = 'Shoe Shop';

      UPDATE ShoeShop
      SET Stock = Stock - 1
      WHERE Product = 'Boots';
ROLLBACK
```

ACID commands

- BEGIN
  - Start the ACID transaction
- COMMIT
  - All statements complete successfully
  - Save the new database state
- ROLLBACK
  - One or more statements fail
  - Undo changes

```
BEGIN
      UPDATE BankAccounts
      SET Balance = Balance - 200
      WHERE AccountName = 'Rose';

      UPDATE BankAccounts
      SET Balance = Balance + 200
      WHERE AccountName = 'Shoe Shop';

      UPDATE ShoeShop
      SET Stock = Stock - 1
      WHERE Product = 'Boots';
ROLLBACK
```

Calling ACID commands

- Can also be issued by some languages
  - Java, C, R, and Python
  - Requires the use of database specific APIs or connectors
- Use the EXEC SQL command to execute SQL statements from code

```
void main ()
 {
   EXEC SQL  UPDATE BankAccounts
           SET Balance = Balance – 200
           WHERE AccountName = 'Rose';
   EXEC SQL  UPDATE BankAccounts
           SET Balance = Balance + 200
           WHERE AccountName = 'Shoe Shop';
  EXEC  SQL  UPDATE  ShoeShop
           SET  Stock  =  Stock  -  1
           WHERE  Product  =  'Boots';
   FINISHED:
   EXEC SQL  COMMIT WORK;
   return;

 SQLERR:
  EXEC SQL  WHENEVER SQLERROR CONTINUE;
  EXEC SQL  ROLLBACK WORK;
  return;
 }
```

Summary

In this video, you learned that:

- A transaction is a unit of work which usually consists of multiple SQL statements
- In an ACID transaction all the SQL statements must complete successfully, or none at all
- ACID stands for Atomic, Consistent, Isolated, Durable
- BEGIN, COMMIT, ROLLBACK
- Can be called from languages like C, R and Python

About this Honors module

Views, Stored Procedures, and Transactions

- **Video:** Views 3 min
- **Ungraded Plugin:** Hands-on Lab: Using Views 10 min
- **Ungraded App Item:** (Optional) Hands-on Lab: Using Views 20 min
- **Video:** Stored Procedures 3 min
- **Ungraded Plugin:** Hands-on Lab: Stored Procedures 10 min
- **Ungraded App Item:** (Optional) Hands-on Lab: Stored Procedures 20 min
- **Video:** ACID Transactions

Summary & Highlights

Congratulations! You have completed this lesson. At this point in the course, you know:

- Views are a dynamic mechanism for presenting data from one or more tables.A transaction represents a complete unit of work, which can be one or more SQL statements.
- An ACID transaction is one where all the SQL statements must complete successfully, or none at all.
- A stored procedure is a set of SQL statements that are stored and executed on the database server, allowing you to send one statement as an alternative to sending multiple statements.
- You can write stored procedures in many different languages like SQL PL, PL/SQL, Java, and C.

Mark as completed

Like Dislike Report an issue

Join Overview

© IBM Corporation. All rights reserved.

Objectives

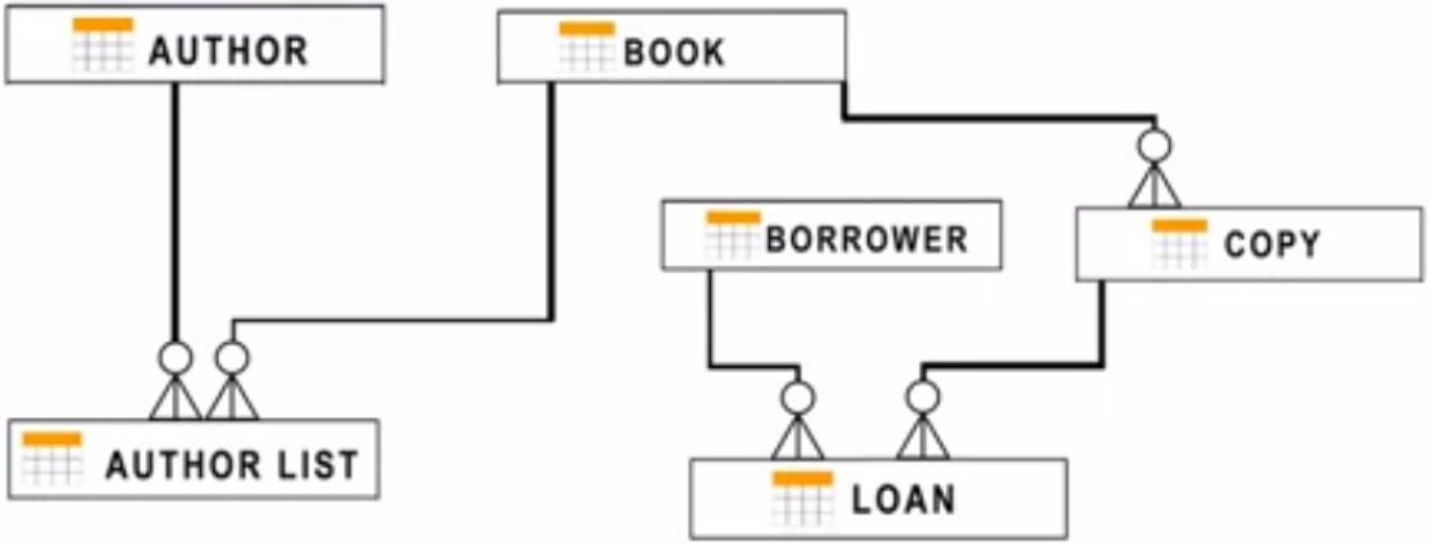
After watching this video, you will be able to:

- Define the join operator
- Explain the role of primary keys and foreign keys in a join operation
- List different types of join operators

Relational model database diagram

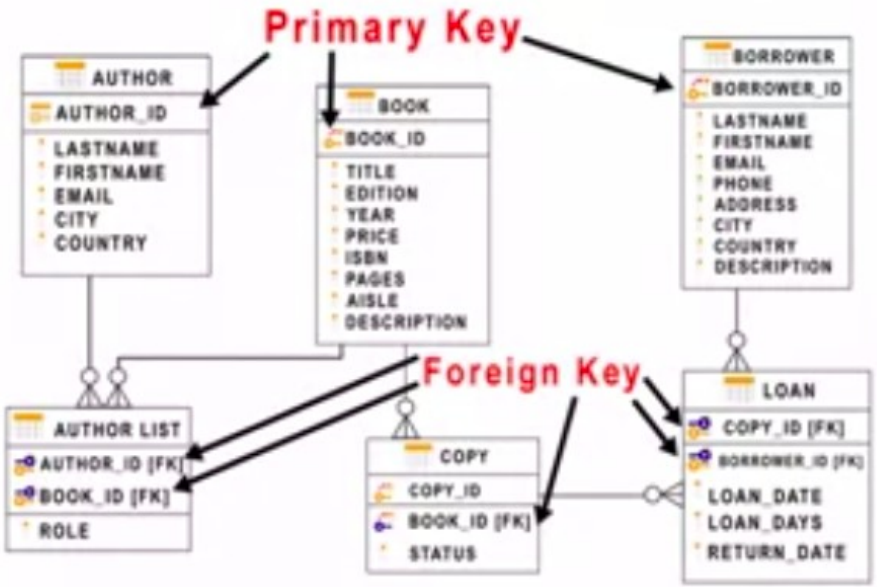
JOIN operator:

- Combines rows from two or more tables
- Based on a relationship



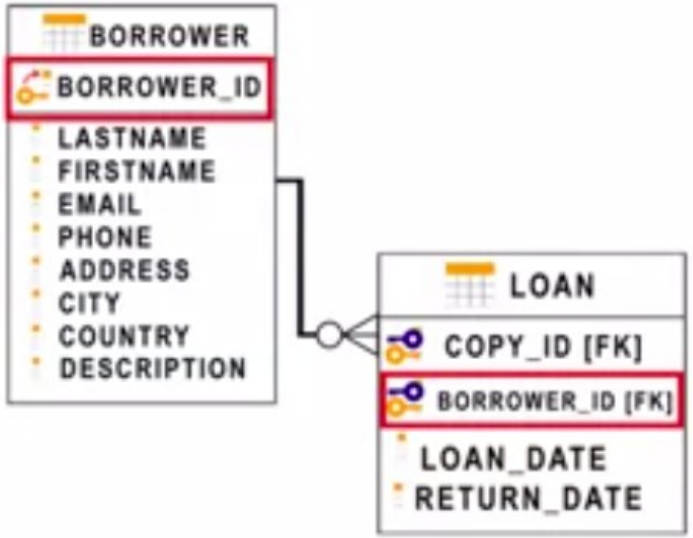
Relational model ER diagram

- Primary key: uniquely identifies each row in a table
- Foreign key: refers to a primary key of another table



Joining tables

Which borrower has a book out on loan?

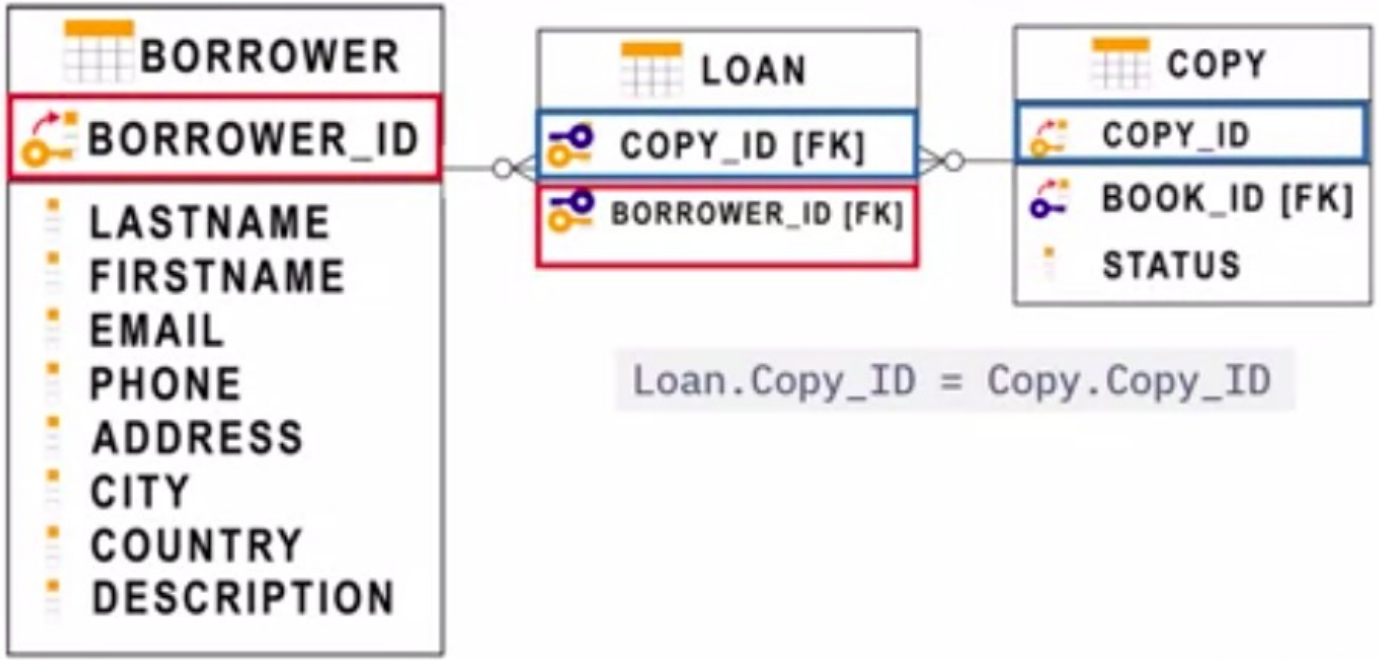



```
Borrower.Borrower_ID = Loan.Borrower_ID
```

Joining Three Tables

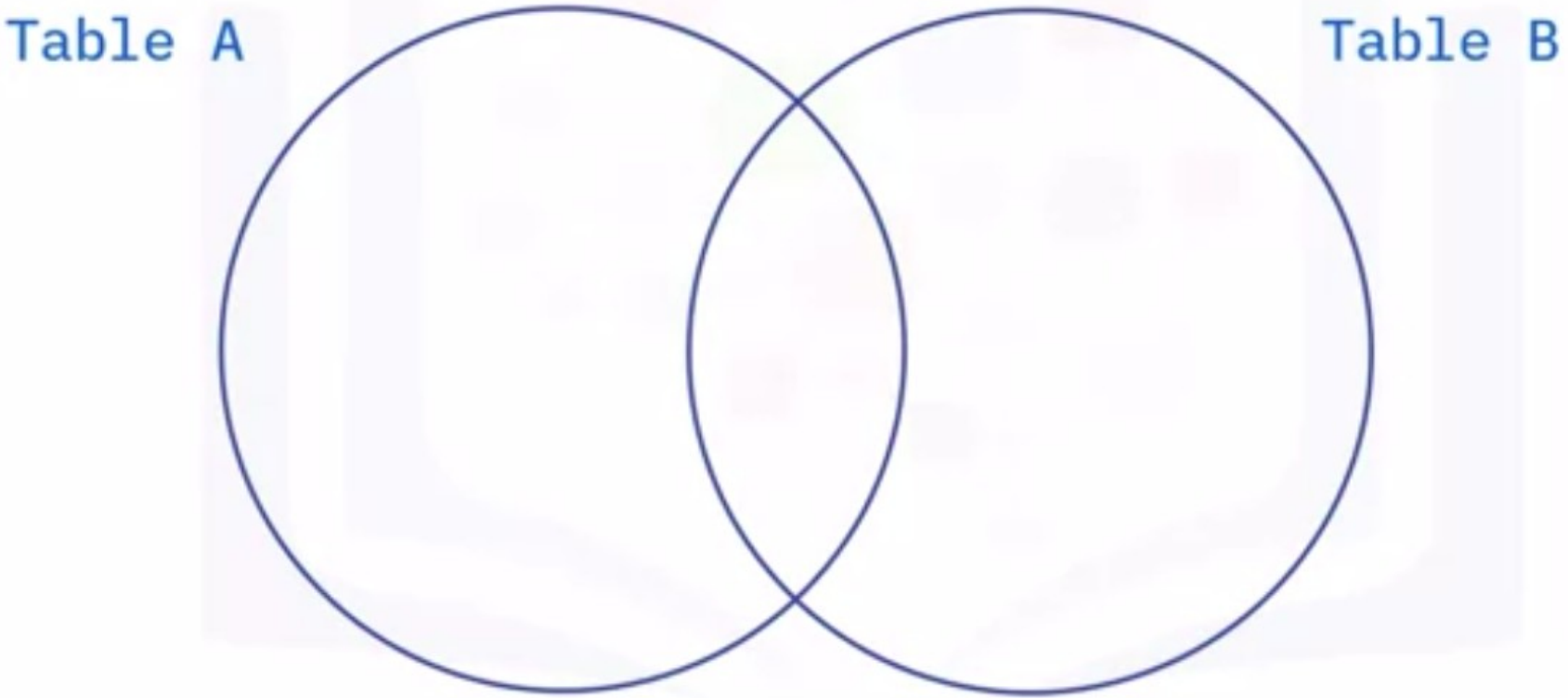
Which copy of a book does the borrower have on loan?

Borrower.Borrower_ID = Loan.Borrower_ID

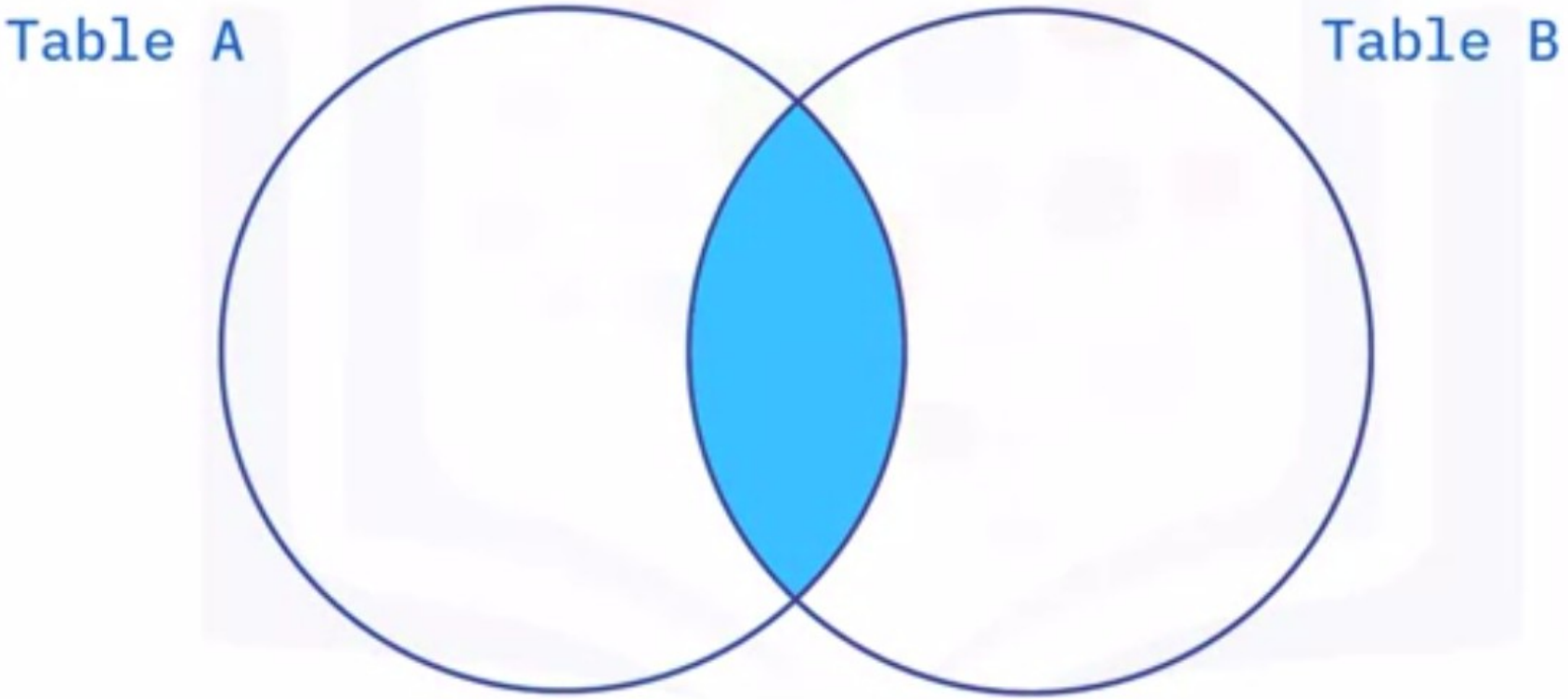


Loan.Copy_ID = Copy.Copy_ID

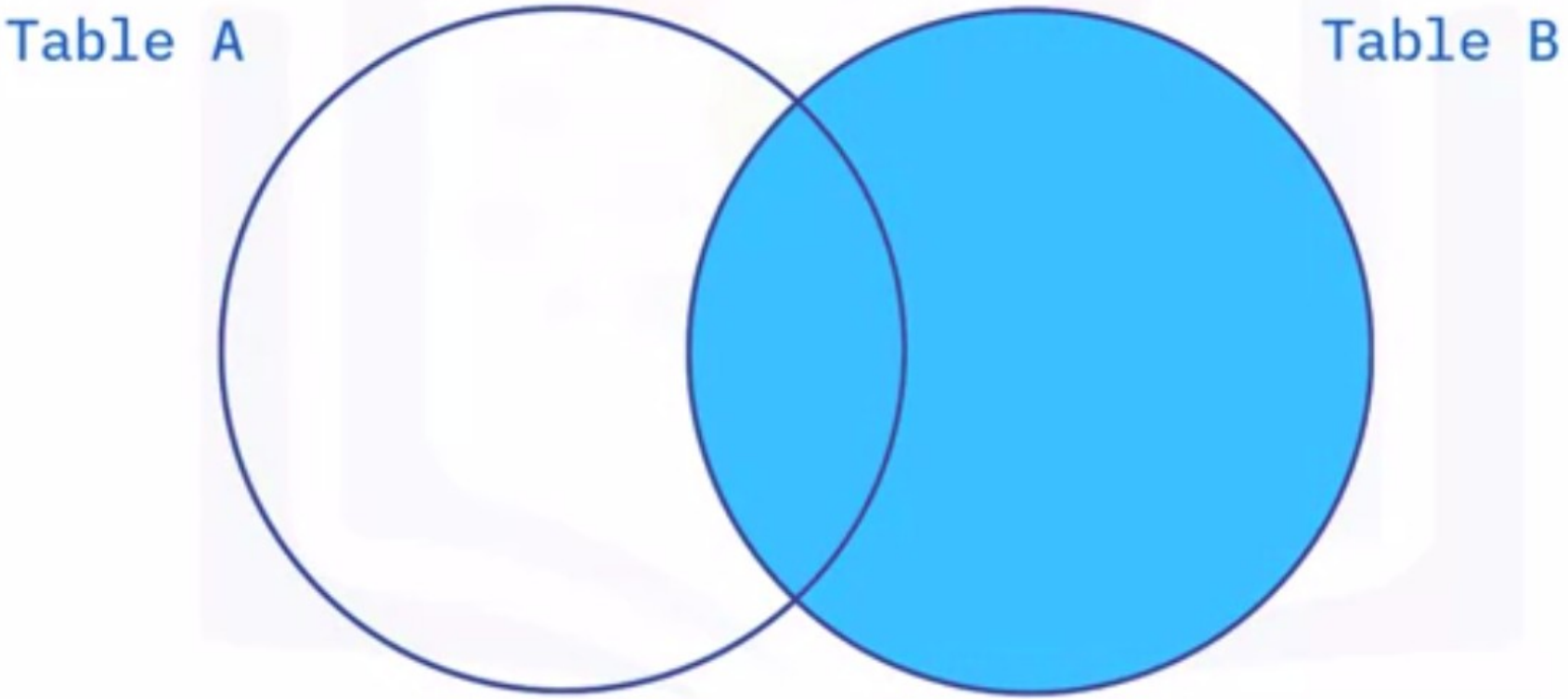
Types of joins



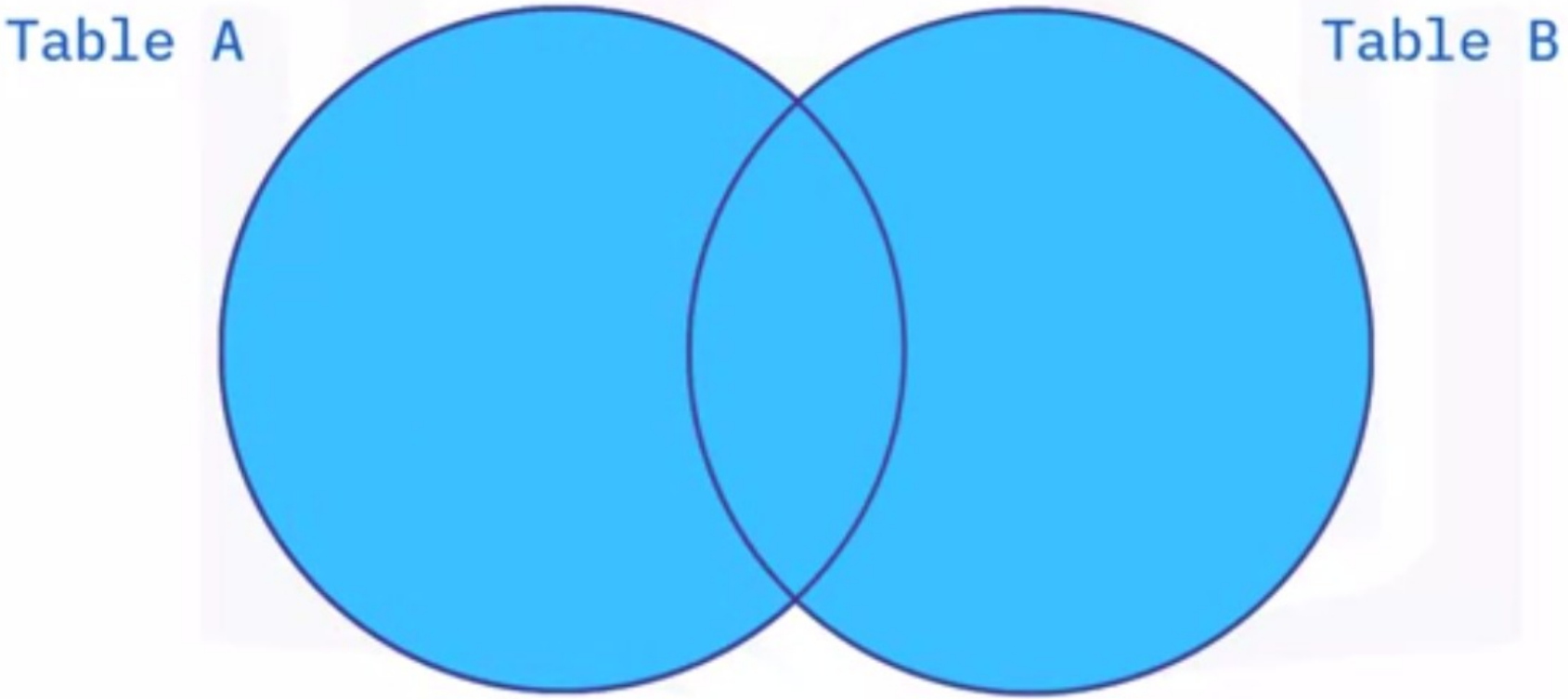
Types of joins



Types of joins



Types of joins



Types of joins

- Inner Join
- Outer Join
  - Left Outer Join
  - Right Outer Join
  - Full Outer Join

Summary

In this video, you learned that:

- The join operator combines rows from two or more tables
- The tables being joined are related through Primary and Foreign keys
- The two types of join are Inner Join and Outer Join

Inner Join

© IBM Corporation. All rights reserved.

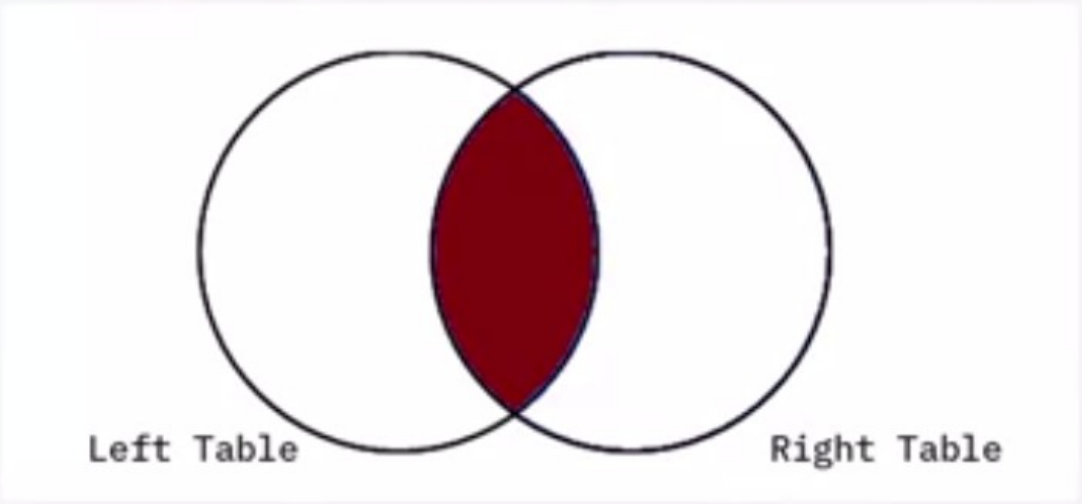
Objectives

After watching this video, you will be able to:

- Describe inner joins
- Explain when to use an inner join
- Describe the syntax of the INNER JOIN statement

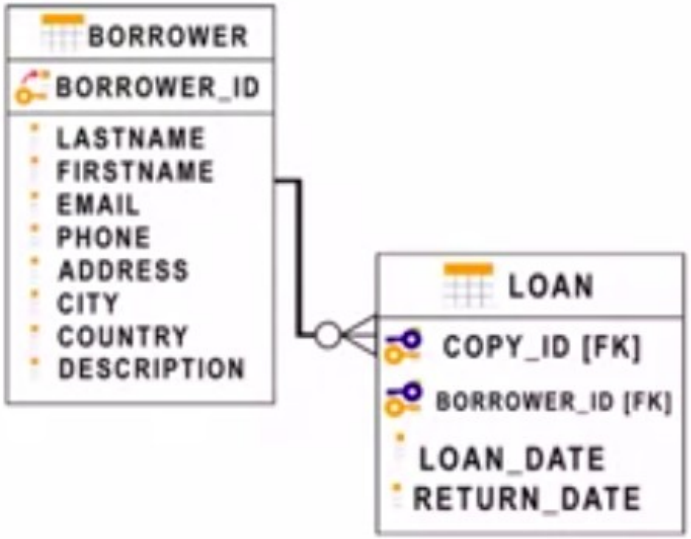
Inner join

- Join operations combine the rows from two or more tables
- Inner join displays matches only



INNER JOIN operator

```
SELECT B.BORROWER_ID, B.LASTNAME, B.COUNTRY,
       L.BORROWER_ID, L.LOAN_DATE
FROM BORROWER B INNER JOIN LOAN L
    ON B.BORROWER_ID = L.BORROWER_ID
```



- In this example, the Borrower table is the Left table
- Each column name is prefixed with an alias to indicate which table each column is associated with

INNER JOIN operator

```
SELECT B.BORROWER_ID, B.LASTNAME, B.COUNTRY,
       L.BORROWER_ID, L.LOAN_DATE
FROM BORROWER B INNER JOIN LOAN L
    ON B.BORROWER_ID = L.BORROWER_ID
```

BORROWER_ID	LASTNAME	COUNTRY
D1	SMITH	CA
D2	SANDLER	CA
D3	SOMMERS	CA
D4	ARDEN	CA
D5	XIE	CA
~~D6~~	~~PETERS~~	~~CA~~
~~D7~~	~~LI~~	~~CA~~
~~D8~~	~~WONG~~	~~CA~~
~~D10~~	~~KIEVA~~	~~CA~~

BORROWER_ID	LOAN_DATE`
D1	11/24/2010
D2	11/24/2010
D3	11/24/2010
D4	11/24/2010
D5	11/24/2010
~~D9~~	~~11/24/2010~~

INNER JOIN operator

```
SELECT B.BORROWER_ID, B.LASTNAME, B.COUNTRY,
      L.BORROWER_ID, L.LOAN_DATE
FROM BORROWER B INNER JOIN LOAN L
    ON B.BORROWER_ID = L.BORROWER_ID
```

BORROWER_ID	LASTNAME	COUNTRY	BORROWER_ID	LOAN_DATE
D1	SMITH	CA	D1	11/24/2010
D2	SANDLER	CA	D2	11/24/2010
D3	SOMMERS	CA	D3	11/24/2010
D4	ARDEN	CA	D4	11/24/2010
D5	XIE	CA	D5	11/24/2010

Summary

In this video, you learned that:

- Inner joins return only rows from joined tables that have a matching value in a common column
- Rows that do not have a matching value do not appear in the result

Outer Join

© IBM Corporation. All rights reserved.

Objectives

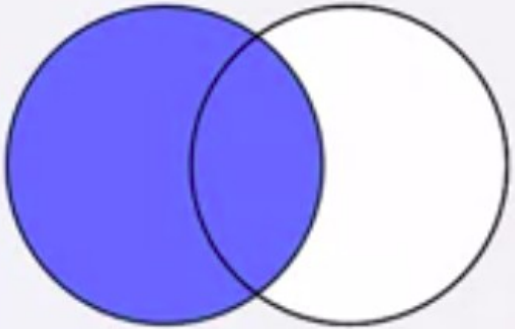
After watching this video, you will be able to:

- Describe left outer joins, right outer joins, and full outer joins
- Explain when to use each type of outer join
- Describe the syntax of the OUTER JOIN statement

Outer joins

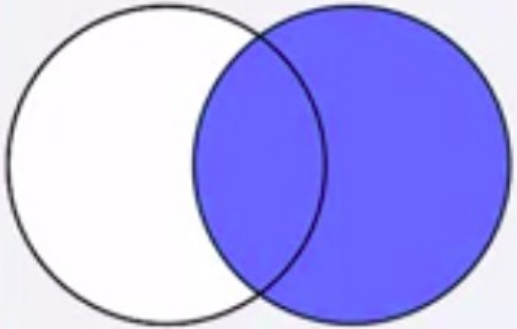
Left Outer Join

All rows from the left table & any matching rows from the right table



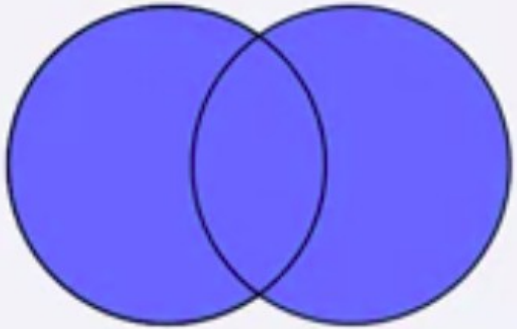
Right Outer Join

All rows from the right table & any matching rows from the left table

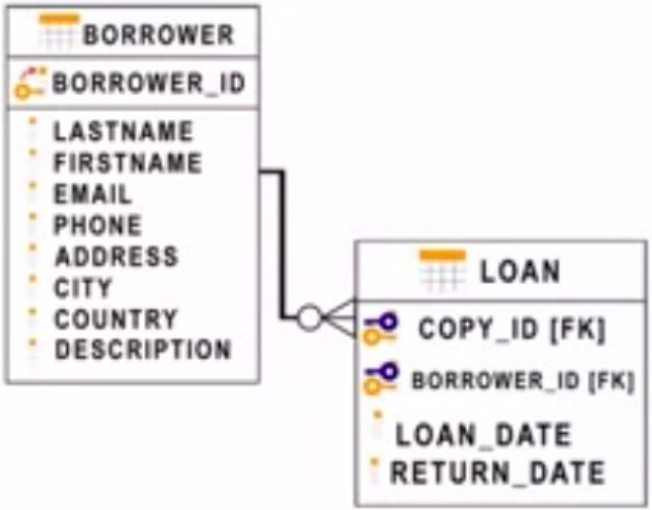


Full Outer Join

All rows from both tables



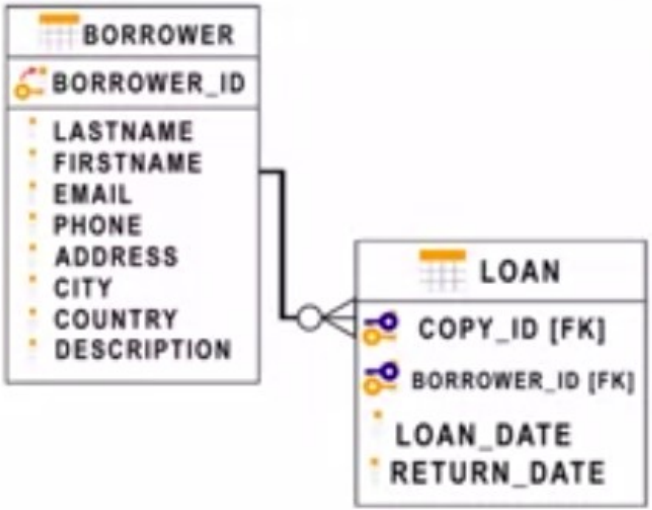
LEFT JOIN Operator




```
SELECT B.BORROWER_ID, B.LASTNAME, B.COUNTRY,
       L.BORROWER_ID, L.LOAN_DATE
FROM BORROWER B LEFT JOIN LOAN L
    ON B.BORROWER_ID = L.BORROWER_ID
```

In this example, the Borrower table is the Left table

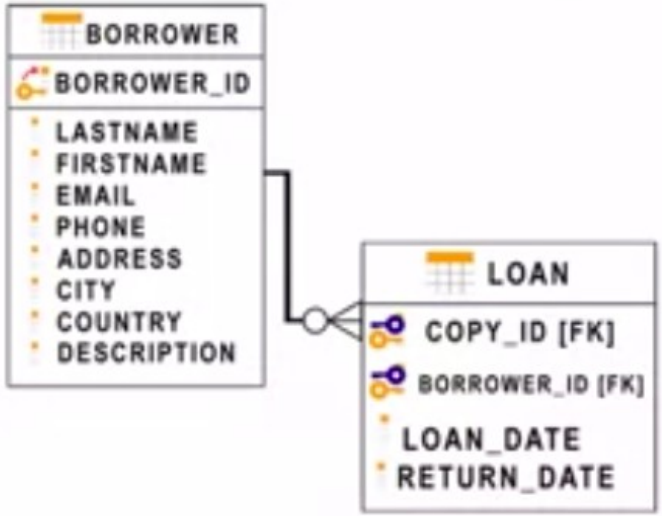
LEFT JOIN Operator




```
SELECT B.BORROWER_ID, B.LASTNAME, B.COUNTRY,
       L.BORROWER_ID, L.LOAN_DATE
FROM BORROWER B LEFT JOIN LOAN L
    ON B.BORROWER_ID = L.BORROWER_ID
```

BORROWER_ID	LASTNAME	COUNTRY	BORROWER_ID	LOAN_DATE
D1	SMITH	CA	D1	11/24/2010
D2	SANDLER	CA	D2	11/24/2010
D3	SOMMERS	CA	D3	11/24/2010
D4	ARDEN	CA	D4	11/24/2010
D5	XIE	CA	D5	11/24/2010
D6	PETERS	CA	NULL	NULL
D7	LI	CA	NULL	NULL
D8	WONG	CA	NULL	NULL

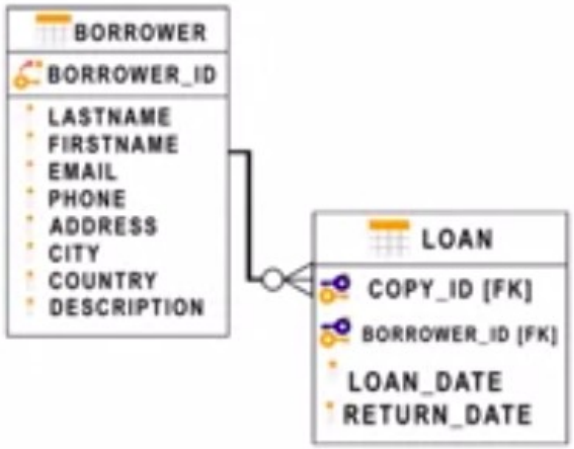
RIGHT JOIN Operator




```
SELECT B.BORROWER_ID, B.LASTNAME, B.COUNTRY,
       L.BORROWER_ID, L.LOAN_DATE
FROM BORROWER B RIGHT JOIN LOAN L
    ON B.BORROWER_ID = L.BORROWER_ID
```

In this example, the Loan table is the right table

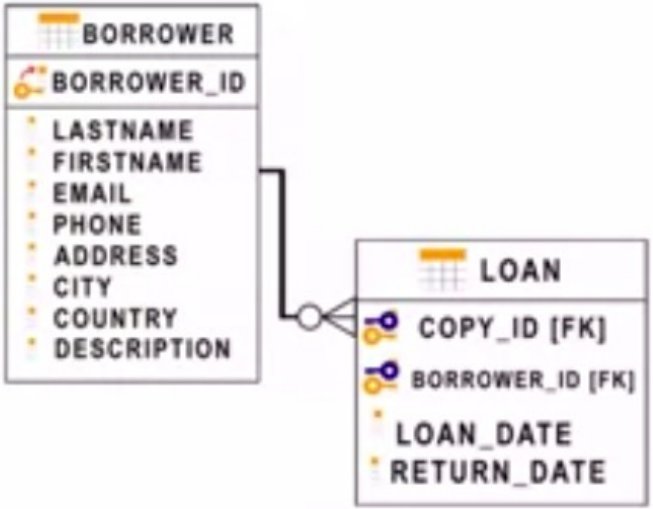
RIGHT JOIN Operator




```
SELECT B.BORROWER_ID, B.LASTNAME, B.COUNTRY,
       L.BORROWER_ID, L.LOAN_DATE
FROM BORROWER B RIGHT JOIN LOAN L
     ON B.BORROWER_ID = L.BORROWER_ID
```

BORROWER_ID	LASTNAME	COUNTRY	BORROWER_ID	LOAN_DATE
D1	SMITH	CA	D1	11/24/2010
D2	SANDLER	CA	D2	11/24/2010
D3	SOMMERS	CA	D3	11/24/2010
D4	ARDEN	CA	D4	11/24/2010
D5	XIE	CA	D5	11/24/2010
NULL	NULL	NULL	D9	11/24/2010

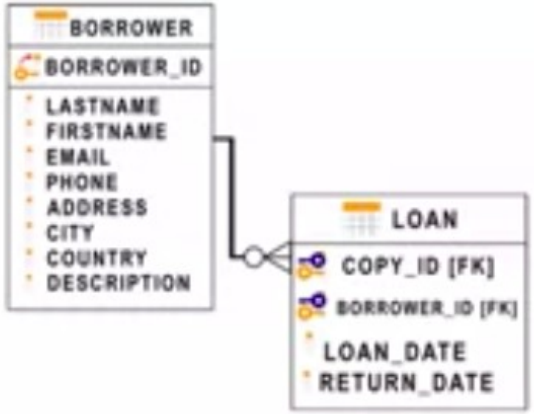
FULL JOIN Operator




```
SELECT B.BORROWER_ID, B.LASTNAME, B.COUNTRY,
       L.BORROWER_ID, L.LOAN_DATE
FROM BORROWER B FULL JOIN LOAN L
    ON B.BORROWER_ID = L.BORROWER_ID
```

In this example, the Borrower table is the Left table

FULL JOIN Operator




```
SELECT B.BORROWER_ID, B.LASTNAME, B.COUNTRY,
       L.BORROWER_ID, L.LOAN_DATE
FROM BORROWER B FULL JOIN LOAN L
    ON B.BORROWER_ID = L.BORROWER_ID
```

BORROWER_ID	LASTNAME	COUNTRY	BORROWER_ID	LOAN_DATE
D1	SMITH	CA	D1	11/24/2010
D2	SANDLER	CA	D2	11/24/2010
D3	SOMMERS	CA	D3	11/24/2010
D4	ARDEN	CA	D4	11/24/2010
D5	XIE	CA	D5	11/24/2010
D6	PETERS	CA	NULL	NULL
D7	LI	CA	NULL	NULL
D8	WONG	CA	NULL	NULL
NULL	NULL	NULL	D9	11/24/2010

Summary

In this video, you learned that:

- Left outer joins return all rows from the left table, and all the rows from the right table that have a match
- Right outer joins return all rows from the right table, and all the rows from the left that have a match
- Full outer joins return all rows from both tables and all the rows from both tables that don't have a match

About this Honors module
Views, Stored Procedures, and Transactions
JOIN Statements
- Video: Join Overview 4 min
- Video: Inner Join 2 min
- Video: Outer Joins 5 min
- Ungraded Plugin: Hands-on Lab: Joins 25 min
- Ungraded App Item: (Optional) Hands-on Lab:Joins 20 min
- Reading: Summary & Highlights 5 min
- Ungraded Plugin: SQL Cheat Sheet: JOIN statements

Summary & Highlights

Congratulations! You have completed this lesson. At this point in the course, you know:

- A join combines the rows from two or more tables based on a relationship between certain columns in these tables.
- To combine data from three or more different tables, you simply add new joins to the SQL statement.
- There are two types of table joins: inner join and outer join; and three types of outer joins: left outer join, right outer join, and full outer join.
- The most common type of join is the inner join, which matches the results from two tables and returns only the rows that match.
- You can use an alias as shorthand for a table or column name.
- You can use a self-join to compare rows within the same table.

Mark as completed

Coursera | Online Courses
Practice Quiz | Coursera
Skills Network Editor
coursera.org/learn/sql-data-science/quiz/mXI06/practice-quiz/attempt?redirectToCover=true
Update
Python Tutor co...
C Program to Ch...
Other bookmarks

Congratulations! You passed!

Grade received 100% **To pass** 70% or higher

Go to next item

1. You usually create a join between…?

1 / 1 point

- ○ Any column in either table.
- ○ The foreign keys in each table.
- ◉ The primary key in one table to the foreign key in another table.
- ○ The primary keys in each table.

Correct
Correct. You usually create a join between a primary key and a foreign key.

2. Which type of join returns all of the rows that an inner join returns and also all of the rows in the second table that do not have a match in the first table?

1 / 1 point

- ○ Full outer join

Back

Practice Quiz

Practice Quiz • 15 min • 5 total points

2. Which type of join returns all of the rows that an inner join returns and also all of the rows in the second table that do not have a match in the first table?

1 / 1 point

- ○ Full outer join
- ○ Left outer join
- ○ Left inner join
- ◉ Right outer join

Correct

Correct. A right outer join returns all of the rows that an inner join returns and all of the rows in the second table that do not have a match in the first table.

3. Which of the following statements correctly uses an INNER JOIN?

1 / 1 point

- ○ CREATE INNER JOIN BETWEEN EMPLOYEES e AND DEPARTMENTS d ON e.DEP_ID = d.DEP_ID
- ◉ SELECT * FROM EMPLOYEES e INNER JOIN DEPARTMENTS d ON e.DEP_ID = d.DEP_ID
- ○ SELECT * FROM EMPLOYEES INNER JOIN DEPARTMENTS ON DEP_ID = DEP_ID

Correct. A right outer join returns all of the rows that an inner join returns and all of the rows in the second table that do not have a match in the first table.

3. Which of the following statements correctly uses an INNER JOIN?

1 / 1 point

- ○ CREATE INNER JOIN BETWEEN EMPLOYEES e AND DEPARTMENTS d ON e.DEP_ID = d.DEP_ID
- ◉ SELECT * FROM EMPLOYEES e INNER JOIN DEPARTMENTS d ON e.DEP_ID = d.DEP_ID
- ○ SELECT * FROM EMPLOYEES INNER JOIN DEPARTMENTS ON DEP_ID = DEP_ID
- ○ SELECT * FROM EMPLOYEES e INNER JOIN DEPARTMENTS d ON DEP_ID

Correct
Correct. This statement uses an inner join to access rows from the EMPLOYEES and DEPARTMENTS tables.

4. Which of the following are three valid types of outer join?

1 / 1 point

- ◉ Left outer join, right outer join, full outer join
- ○ Left outer join, right outer join, total outer join
- ○ Left outer join, right outer join, both outer join

Back

Practice Quiz

Practice Quiz • 15 min • 5 total points

- SELECT * FROM EMPLOYEES INNER JOIN DEPARTMENTS ON DEP_ID = DEP_ID
- SELECT * FROM EMPLOYEES e INNER JOIN DEPARTMENTS d ON DEP_ID

Correct

Correct. This statement uses an inner join to access rows from the EMPLOYEES and DEPARTMENTS tables.

4. Which of the following are three valid types of outer join?

1 / 1 point

- (●) Left outer join, right outer join, full outer join
- ( ) Left outer join, right outer join, total outer join
- ( ) Left outer join, right outer join, both outer join
- ( ) Left outer join, right outer join, left/right outer join

Correct

Correct. These are three types of outer join.

5. Which type of join would you use to select all the rows from both tables?

1 / 1 point

- Left outer join, right outer join, total outer join
- Left outer join, right outer join, both outer join
- Left outer join, right outer join, left/right outer join

Correct
Correct. These are three types of outer join.

5. Which type of join would you use to select all the rows from both tables?

1 / 1 point

- Right outer join
- Left outer join
- Total outer join
- Full outer join

Correct
Correct. A full outer join will return all the rows from both the left table and the right table in the join.